Supplementary Materials of "NetGO: Improving Large-scale Protein Function Prediction with Massive Network Information"

Ronghu You[1,2,3], Shuwei Yao[1,2,3], Yi Xiong[4], Xiaodi Huang[5], Fengzhu Sun[2,3,6], Hiroshi Mamitsuka[7], Shanfeng Zhu[1,2,3*]

[1] School of Computer Science and Shanghai Key Lab of Intelligent Information Processing and

[2] Institute of Science and Technology for Brain-Inspired Intelligence, Fudan University, Shanghai 200433, China,

[3] Key Laboratory of Computational Neuroscience and Brain-Inspired Intelligence (Fudan University), Ministry of Education, China

[4] Department of Bioinformatics and Biostatistics, Shanghai Jiao Tong University,

[5] School of Computing and Mathematics, Charles Sturt University, Albury, NSW 2640, Australia,

[6] Department of Biological Sciences, University of Southern California, Los Angeles, CA 90089, USA,

[7] Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji 611-0011, Japan

1 Background

The official result of CAFA3 was announced in the Function SIG meeting of ISMB2018 (Chicago, July 2018) [1], which used new experimentally annotated proteins between Feb 2017 and Nov 2017 as the test data. From the announcements of the meetings, GOLabeler achieved the first place for *no-knowledge* proteins out of around 150 submissions all over the world in terms of F-max in all three GO ontologies: MFO, BPO, and CCO (CAFA3 assessor: N. Zhou). Specifically, GOLabeler had achieved 0.62 for MFO, which was more than 14%, higher than other methods. On the other hand, GOLabeler achieved 0.40 and 0.61 for BPO and CCO, respectively. But the improvement over other methods is only slight. This motivated us to develop NetGO. NetGO incorporates massive network information into GOLabeler so as to improve the prediction performance in BPO and CCO.

2 Methods

NetGO consists of six component methods, Naive, BLAST-KNN, LR-InterPro, LR-3mer, LR-ProFET and Net-KNN. The details of the top five component method have been presented in [1]. Here we give the formula of Net-KNN.

*Corresponding author: zhusf@fudan.edu.cn

[1] https://www.biofunctionprediction.org/afp_programs/AFP-2018-program.pdf

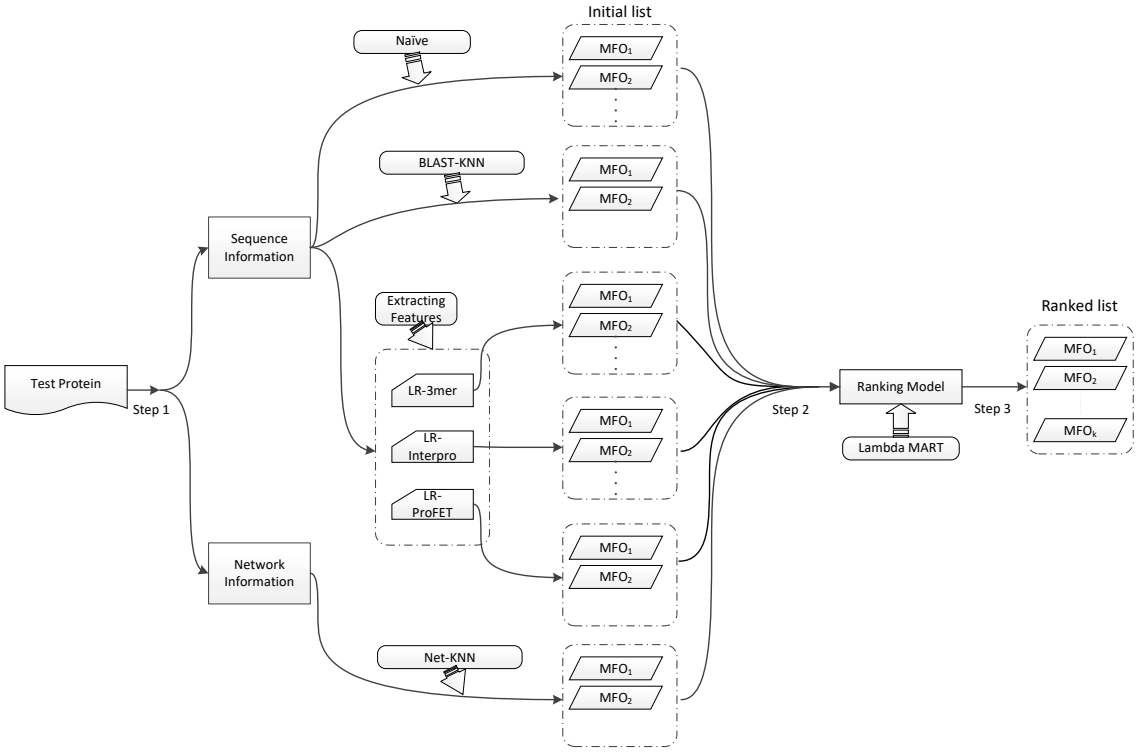

Figure 1: The workflow of a test process in NetGO.

2.1 Net-KNN

Given a test protein P_j and a protein network (from STRING in our experiments), Net-KNN computes the score $S(G_i, P_j)$ between P_j and GO term G_i, through the following equation:

$$S(G_i, P_j) = \frac{\sum_{PV_k \in PV} I(G_i, PV_k) * \omega(PV_j, PV_k)}{\sum_{PV_k \in PV} \omega(PV_j, PV_k)}, \tag{1}$$

where PV_j is the node in the protein protein network corresponding to P_j by STRI or HOMO (see the main text for the details).

Given that there are m networks $PN^{(l)}$ $(l = 1, \cdots, m)$ over the same set of nodes, the aggregated weight $\omega(PV_j, PV_k)$ can be computed in an ensemble way:

$$\omega(PV_i, PV_j) = 1 - \prod_{l=1}^{m} (1 - \omega^{(l)}(PV_i, PV_j)) \tag{2}$$

where again $\omega^{(l)}(PV_i, PV_j)$ is the confidence of an association between PV_i and PV_j in $PN^{(l)}$, and $\omega(PV_i, PV_j) = 0$ if there are no direct edges between PV_i and PV_j.

The higher the weight of two proteins in all of the m individual networks is, the higher their aggregated weight is.

2.2 The workflow of processing queries in NetGO

Fig. 1 illustrates the workflow of NetGO during testing.

Step 1: Generate candidates GO terms

Given a query protein, we run six component methods to produce the top-k GO terms from each component and then merge them to form the candidate GO terms (we used k=30 in our experiments.).

Table 1: #proteins of species that have at least ten proteins under each of three GO ontologies in the datasets of Training, LTR1, LTR2, and Testing, respectively.

Species	Training			LTR1			LTR2			Testing		
	MFO	BPO	CCO	MFO	BPO	CCO	MFO	BPO	CCO	MFO	BPO	CCO
HUMAN (*Homo sapiens*)	9,487	11,531	18,417	83	97	336	1,427	332	193	150	228	1,202
MOUSE (*Mus musculus*)	5,929	9,497	9,029	126	281	173	118	166	97	63	176	145
DROME (*Drosophila melanogaster*)	4,716	10,480	7,739	179	669	282	171	158	167	76	113	79
ARATH (*Arabidopsis thaliana*)	4,253	8,135	10,004	206	241	180	476	155	87	111	257	186
DANRE (*Danio rerio*)	2,308	9,104	1,694	77	529	38	55	68	40	73	645	41
RAT (*Rattus norvegicus*)	4,330	5,329	4,684	43	129	134	44	52	45	40	85	84
All species (not only the above)	48,453	81,869	78,186	805	2,060	1,340	2,531	1,089	745	662	1,758	1,852

Note that reducing k is to focus on the most relevant GO terms to the query protein and also reduce the computational burden of the model.

Step 2: Generate features for ranking GO terms

We then generate features of the query protein by using the scores (of each of the candidate GO terms) predicted by all six component methods. As such, a six-dimensional feature vector is formed for each pair of a GO term and one query protein. All score values are between 0 and 1.

Step 3: Rank GO terms by learning to rank (LTR)

Finally, we use LTR to rank all candidate GO terms of each query protein. All proteins in the training data and their candidate GO terms are used for training the LTR model. In this way, LTR integrates multiple sequence- and network-based evidence of proteins.

3 Experimental settings

3.1 Network information in STRING

We have used six different types of protein protein networks in STRING database: 0:neighbourhood, 1:fusion, 2:co-occurrence, 3:coexpression, 4:experiment and 5:database. The neighbourhood network is constructed by linking genes that occur very closely in genomes. The co-occurrence network is derived from phylogenetic profile. The fusion network is constructed by considering fusion genes per species. The co-expression network is derived from mRNA expression data. The experiment network is derived from experimental data. The database network is derived from curated databases.

3.2 Benchmark Datasets

We separated test data from training data in the same time-series way as CAFA did. Also, we used the same target species as CAFA3 in LTR1, LTR2 and Testing. Table 1 reports the number of proteins in the above four datasets. Note that the inputs of GOLabeler are sequences only, while those of NetGO include both sequence and network information.

3.3 Performance Evaluation Metrics

We used the three measures for performance evaluation: F_{max}, S_{min}, and AUPR (Area Under the Precision-Recall curve). As a standard evaluation metric in machine learning, AUPR punishes false positive prediction. It is suitable for highly imbalanced data. F_{max} is an official metric of CAFA with the following definition.

$$F_{max} = \max_{\tau} \left\{ \frac{2 \cdot \mathrm{pr}(\tau) \cdot \mathrm{rc}(\tau)}{\mathrm{pr}(\tau) + \mathrm{rc}(\tau)} \right\}, \tag{3}$$

where $\mathrm{pr}(\tau)$ and $\mathrm{rc}(\tau)$ are *precision* and *recall*, respectively, obtained at some cut-off value τ, defined as follows, respectively:

$$\mathrm{pr}(\tau) = \frac{1}{h(\tau)} \sum_{j=1}^{h(\tau)} \frac{\sum_i \mathbf{1}(S(G_i, P_j) \geq \tau) \cdot I(G_i, P_j)}{\sum_i \mathbf{1}(S(G_i, P_j) \geq \tau)}. \quad (4)$$

$$\mathrm{rc}(\tau) = \frac{1}{N_T} \sum_{j=1}^{N_T} \frac{\sum_i \mathbf{1}(S(G_i, P_j) \geq \tau) \cdot I(G_i, P_j)}{\sum_i I(G_i, P_j)}, \quad (5)$$

The *remaining uncertainty* (ru), *misinformation* (mi), and the resulting *minimum semantic distance* (S_{min}) are defined as:

$$ru(\tau) = \frac{1}{n_e} \sum_{i=1}^{n_e} \sum_f ic(f) \cdot \mathbf{1}\left(f \notin P_i(\tau) \wedge f \in T_i\right) \quad (6)$$

$$mi(\tau) = \frac{1}{n_e} \sum_{i=1}^{n_e} \sum_f ic(f) \cdot \mathbf{1}\left(f \in P_i(\tau) \wedge f \notin T_i\right) \quad (7)$$

$$S_{\min} = \min_{\tau} \left\{ \sqrt{ru(\tau)^2 + mi(\tau)^2} \right\} \quad (8)$$

where $ic(f)$ is the pre-calculated *information content.*

3.4 The setting of Component Methods and LTR

1. BLAST-KNN
   We used blast ver. 2.3.0+ with default parameters for BLAST-KNN (), except that the number of iterations was one, blastdb was from all proteins in D and the E-value cutoff was set to 0.001.

2. LR
   LR classifiers were trained by using sklearn with default parameters. As such, regularization was applied with regularization strength set as 1.0.

3. LTR
   We used 'rank:pairwise' as the objective loss function in xgboost. Also the maximum depth of trees in MART (Multiple Additive Regression Trees) was set at 3, to avoid overfitting to the training data.

3.5 The Selection of k

For training NetGO, we combined top k GO terms by each component method as the candidate GO terms. For choosing a suitable k, we conducted 5-fold cross validation over LTR training set with k=10, 30, 50, 70. Table 2 reports the performance. We found that the best F_{max} is achieved when k=30 . With a large value of k, the performance will decrease slightly and it will take much more time to train the ranking model. Considering all of these facts, we therefore set k=30 in NetGO.

Table 2: The performance of NetGO over LTR training by 5-fold cross validation with different settings of k.

k	F_{max}			AUPR		
	MFO	BPO	CCO	MFO	BPO	CCO
10	0.659	**0.349**	0.668	0.656	0.226	0.680
30	**0.662**	**0.349**	**0.672**	**0.658**	0.226	**0.695**
50	0.660	0.347	0.670	0.651	0.226	**0.695**
70	0.658	0.348	0.669	0.653	**0.228**	0.694

4 Experimental results

4.1 AUPR of Net-KNN using different types of networks

Table 3: The AUPR scores of Net-KNN as a result of using six different types of networks: from 0:neighbourhood to 5:database. "All (binary, cut-off=0.5 or 0.9)": a combination (union) of all networks, where the weight of each network edge (similarity) is transformed into a binary by using the cut-off value of 0.5 or 0.9; and "All": a combination of all networks.

	All species			Eukaryote			Prokaryote		
	MFO	BPO	CCO	MFO	BPO	CCO	MFO	BPO	CCO
0: neighbourhood	0.047	0.023	0.195	0.052	0.230	0.195	*0.030*	0.025*	**0.197***
1: fusion	0.067	0.013	0.139	0.083	0.012	0.138	0.020	0.016	0.020
2: cooccurence	0.062	0.013	0.128	0.072	0.013	0.130	0.026	*0.022*	*0.160*
3: coexpression	*0.111*	0.073*	*0.475*	*0.132*	0.076*	*0.478*	0.027	*0.022*	0.124
4: experiment	0.101	*0.066*	0.504*	0.125	*0.071*	0.509*	0.033*	0.020	0.023
5: database	0.128*	0.048	0.366	0.150*	0.048	0.363	0.021	0.019	0.004
All (binary, cut-off=0.5)	0.119	0.053	0.481	0.143	0.056	0.482	0.028	0.015	0.134
All (binary, cut-off=0.9)	0.148	0.040	0.413	0.170	0.041	0.416	**0.044**	0.025	0.065
All	**0.158**	**0.097**	**0.568**	**0.186**	**0.102**	**0.567**	0.042	**0.035**	0.166

We examined AUPR of Net-KNN by using six different types of networks in STRING: 0:neighbourhood, 1:fusion, 2:co-occurrence, 3:coexpression, 4:experiment, and 5:database. Table 3 reports AUPR obtained by using each individual network (the upper part) and by combining all of these networks (the lower part). The best performance values are highlighted in boldface. "All" in the table means the combination of all the networks, according to equation (2). In the upper part of the table, the highest and second highest scores are marked with an asterisk and are displayed in italic, respectively.

We have three main findings:

1) The use of all networks ("All" or "All(binary, cut-off=0.9)" in Table 3) achieved the best performance, except for CCO of prokaryotes.

2) Among the six networks, no one performed the best in more than three cases. This implies that the role of each network is different from each other. For example, 0:neighbourhood and 2:cooccurrence achieved the best and second for CCO of prokaryote, respectively. However, they are ranked as the 4th and 6th for CCO of eukaryote, respectively. This indicates that they are only useful for predicting prokaryote proteins.

3) Among three settings of network combinations, keeping the association scores of network edge ("All" in Table 3) achieved the best performance in all nine cases, except for MFO of Prokaryote. This implies that the conversion of the association scores into binary by using a cut-off value will cause information loss. So we use "All" as a component of NetGO.

4.2 Performance comparisons of NetGO with other competing methods with the $p-$values

To validate the performance of NetGO and other competing methods, we resampled the instances in testing with replacement 100 times (bootstrap with replacement) to make the experiment reliable. We get 100 datasets that have the same number of proteins as the test set. Except for the performance evaluation measures, we used the paired t-test to statistically evaluate the performance difference between the best performance (in boldface in tables) and all others. The result was considered significant if p-value was smaller than 0.05. As shown in Table 4, the best performance value is underlined if the value

Table 4: Performance comparisons of NetGO with other competing methods using 100 bootstrapped datasets with replacement. We use a paired t-test to statistically evaluate the performance difference between the best method (in boldface in tables) and all other methods.

	F_{max}			AUPR		
	MFO	BPO	CCO	MFO	BPO	CCO
Naive	0.318	0.255	0.604	0.170	0.116	0.611
	3.29e-131	5.06e-123	2.92e-126	6.12e-129	3.75e-114	1.38e-128
BLAST-KNN	0.591	0.284	0.642	0.449	0.112	0.562
	9.18e-85	8.33e-122	7.70e-87	1.84e-91	2.58e-132	1.17e-126
Net-KNN	0.346	0.306	0.642	0.159	0.096	0.566
	3.55e-130	2.51e-84	3.32e-95	1.90e-133	5.51e-126	2.55e-121
DeepGO	0.381	0.243	0.569	0.241	0.092	0.536
	3.93e-132	5.94e-130	4.25e-131	6.90e-127	3.50e-124	1.77e-131
GoFDR	0.543	0.272	0.571	0.334	0.065	0.328
	7.11e-94	9.96e-114	1.95e-134	8.48e-113	6.92e-140	5.11e-172
GOLabeler	0.630	0.321	0.668	0.549	0.171	0.685
	5.45e-09	5.95e-98	3.60e-60	2.79e-49	1.13e-100	1.74e-102
NetGO(LTR1)	0.630	0.339	0.669	0.546	0.191	0.698
	1.37e-11	3.38e-37	7.71e-78	8.62e-55	3.65e-57	7.50e-127
NetGO(LTR1+2)	**0.632**	**0.342**	**0.674**	**0.556**	**0.196**	**0.708**

is statistically significant. In the lower part, we compared two variants of NetGO. NetGO(LTR1) used LTR1 only to train the ranking model, while NetGO(LTR1+2) used both LTR1 and LTR2. We can see that using both LTR1 and LTR2 achieved the best performance, which is consistent with the case in GOLabeler [1]. To make full use of massive network information, we show the results of NetGO by using both LTR1 and LTR2 that incorporates all 6 types of networks in STRING.

4.3 Performance comparisons over testing data in terms of $S_{\min}$

Table 5 reports the performance of NetGO and its competing methods in terms of $S_{\min}$.

Table 5: Performance comparisons over testing data in terms of$S_{\min}$

	MFO	BPO	CCO
Naive	9.153	16.029	5.266
BLAST-KNN	6.519	16.069	4.990
Net-KNN	9.061	16.525	4.961
DeepGO	8.461	16.334	5.244
GoFDR	7.681	31.395	10.929
GOLabeler	6.219	15.582	4.769
NetGO	**6.170**	**15.270**	**4.647**

4.4 Precision-recall curves of NetGO with its competing methods over MFO and CCO.

As shown in Figs 2 and 3, we present the Precision-recall curves of NetGO with its own components and competing methods over MFO and CCO, respectively.

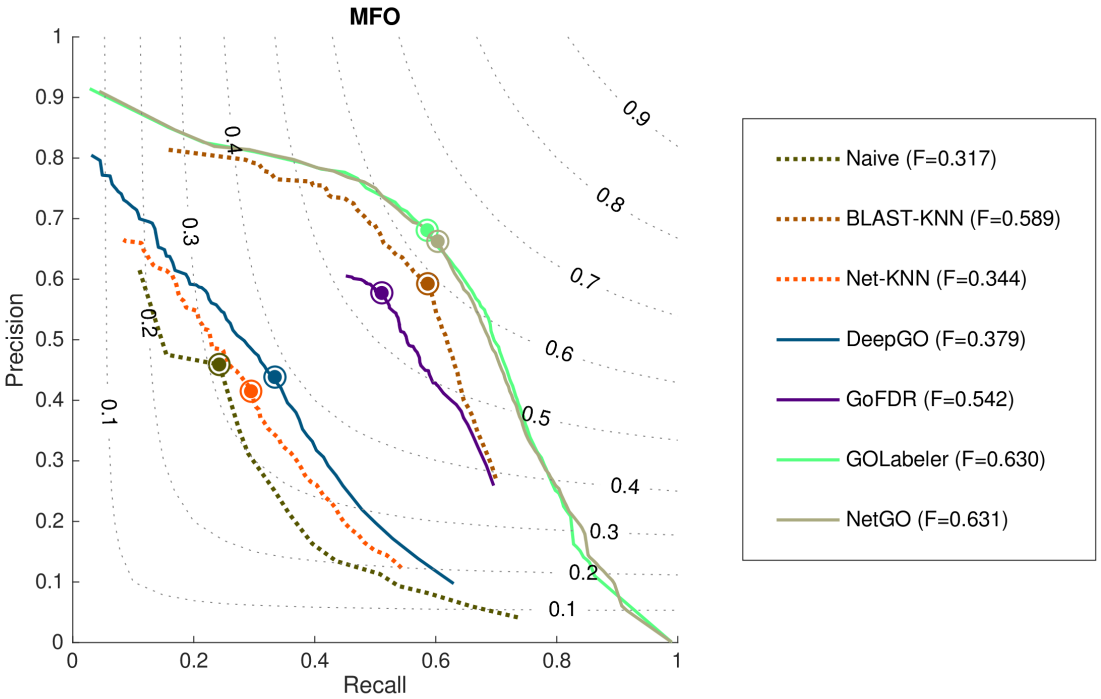

Figure 2: Precision-recall curves of NetGO with its own components and competing methods over MFO.

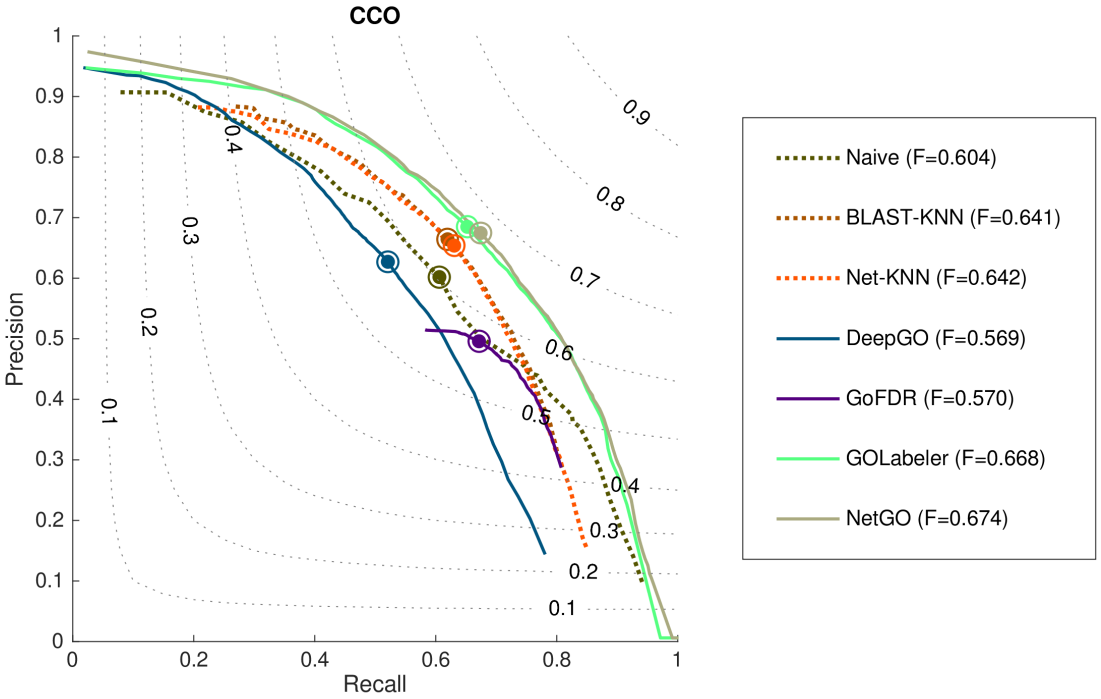

Figure 3: Precision-recall curves of NetGO with its own component and competing methods over CCO.

4.5 Running Time

The time cost for the NetGO web server depends on the number of input data points. Table 6 lists the average running times of 10 randomly generated datasets for each protein in the testing set.

Table 6: Mean running time of NetGO web server for different number of input proteins

Protein num	GOLabeler	NetGO
1	204.72s	230.93s
100	914.85s	997.12s
200	1502.87s	1573.13s
400	2707.82s	3094.76s
1000	4879.79s	5538.76s

References

[1] You,R., Zhang,Z., Xiong,Y., Sun,F., Mamitsuka,H., and Zhu,S. (2018) GOLabeler: improving sequence-based large-scale protein function prediction by learning to rank. *Bioinformatics,* **34**(14), 2465–2473 [PubMed:29522145] [doi:10.1093/bioinformatics/bty130].